

Planning and Optimization

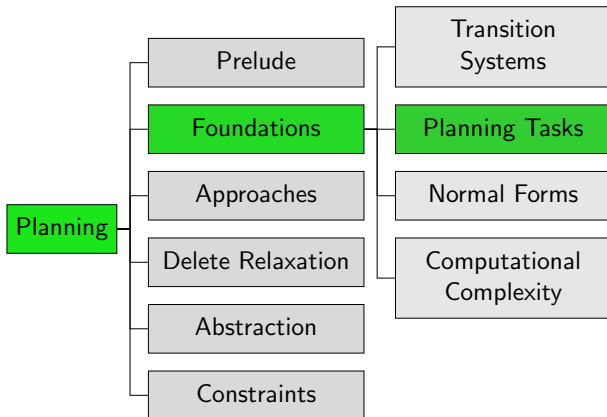
B3. Formal Definition of Planning

Malte Helmert and Gabriele Röger

Universität Basel

September 23, 2026

Content of the Course



Semantics of Effects and Operators

Semantics of Effects: Trigger Conditions

Definition (Trigger Condition for an Effect)

Let ℓ be an atomic effect, and let e be an effect.

The **trigger condition** or **trigger** $\text{trig}(\ell, e)$ for the firing of ℓ given the effect e is a propositional formula defined as follows:

- $\text{trig}(\ell, \top) = \perp$
- $\text{trig}(\ell, e) = \top$ for the atomic effect $e = \ell$
- $\text{trig}(\ell, e) = \perp$ for all atomic effects $e = \ell' \neq \ell$
- $\text{trig}(\ell, (e \wedge e')) = (\text{trig}(\ell, e) \vee \text{trig}(\ell, e'))$
- $\text{trig}(\ell, (\chi \triangleright e)) = (\chi \wedge \text{trig}(\ell, e))$

Intuition: $\text{trig}(\ell, e)$ represents the condition that must be true in the current state for the effect e to lead to the atomic effect ℓ

Trigger Condition: Example (1)

Example

Consider the move operator m_1 from the running example:

$$\text{eff}(m_1) = ((t_1 \triangleright \neg t_1) \wedge (\neg t_1 \triangleright t_1)).$$

Under which conditions does it set t_1 to false?

$$\begin{aligned} \text{trig}(\neg t_1, \text{eff}(m_1)) &= \text{trig}(\neg t_1, ((t_1 \triangleright \neg t_1) \wedge (\neg t_1 \triangleright t_1))) \\ &= \text{trig}(\neg t_1, (t_1 \triangleright \neg t_1)) \vee \\ &\quad \text{trig}(\neg t_1, (\neg t_1 \triangleright t_1)) \\ &= (t_1 \wedge \text{trig}(\neg t_1, \neg t_1)) \vee \\ &\quad (\neg t_1 \wedge \text{trig}(\neg t_1, t_1)) \\ &= (t_1 \wedge \top) \vee (\neg t_1 \wedge \perp) \\ &\equiv t_1 \vee \perp \\ &\equiv t_1 \end{aligned}$$

Trigger Condition: Example (2)

Example

Consider the move operator m_1 from the running example:

$$\text{eff}(m_1) = ((t_1 \triangleright \neg t_1) \wedge (\neg t_1 \triangleright t_1)).$$

Under which conditions does it set i to true?

$$\begin{aligned} \text{trig}(i, \text{eff}(m_1)) &= \text{trig}(i, ((t_1 \triangleright \neg t_1) \wedge (\neg t_1 \triangleright t_1))) \\ &= \text{trig}(i, (t_1 \triangleright \neg t_1)) \vee \\ &\quad \text{trig}(i, (\neg t_1 \triangleright t_1)) \\ &= (t_1 \wedge \text{trig}(i, \neg t_1)) \vee \\ &\quad (\neg t_1 \wedge \text{trig}(i, t_1)) \\ &= (t_1 \wedge \perp) \vee (\neg t_1 \wedge \perp) \\ &\equiv \perp \vee \perp \\ &\equiv \perp \end{aligned}$$

Trigger Condition: Example (3)

Example

Consider the unload operator u from the running example:

$$\begin{aligned} \text{eff}(u) = & \neg i \wedge (w \triangleright ((t_1 \triangleright w) \wedge (\neg t_1 \triangleright \neg w))) \\ & \wedge (\neg w \triangleright ((t_2 \triangleright w) \wedge (\neg t_2 \triangleright \neg w))). \end{aligned}$$

Under which conditions does it set w to true?

↪ exercises

Semantics of Effects: Applying an Effect

first attempt:

Definition (Applying Effects)

Let V be a set of propositional state variables.

Let s be a state over V , and let e be an effect over V .

The **resulting state** of applying e in s , written $s[e]$, is the state s' defined as follows for all $v \in V$:

$$s'(v) = \begin{cases} \mathbf{T} & \text{if } s \models \text{trig}(v, e) \\ \mathbf{F} & \text{if } s \models \text{trig}(\neg v, e) \\ s(v) & \text{otherwise} \end{cases}$$

What is the problem with this definition?

Semantics of Effects: Applying an Effect

correct definition:

Definition (Applying Effects)

Let V be a set of propositional state variables.

Let s be a state over V , and let e be an effect over V .

The **resulting state** of applying e in s , written $s[e]$, is the state s' defined as follows for all $v \in V$:

$$s'(v) = \begin{cases} \mathbf{T} & \text{if } s \models \text{trig}(v, e) \\ \mathbf{F} & \text{if } s \models \text{trig}(\neg v, e) \wedge \neg \text{trig}(v, e) \\ s(v) & \text{otherwise} \end{cases}$$

Add-after-Delete Semantics

Note:

- The definition implies that if a variable is simultaneously “added” (set to **T**) and “deleted” (set to **F**), the value **T** takes precedence.
- This is called **add-after-delete semantics**.
- This detail of effect semantics is somewhat arbitrary, but has proven useful in practice.

Semantics of Operators

Definition (Applicable, Applying Operators, Resulting State)

Let V be a set of propositional state variables.

Let s be a state over V , and let o be an operator over V .

Operator o is **applicable** in s if $s \models \text{pre}(o)$.

If o is applicable in s , the **resulting state** of **applying** o in s , written $s[o]$, is the state $s[\text{eff}(o)]$.

Planning Tasks

Planning Tasks

Definition (Planning Task)

A (propositional) **planning task** is a 4-tuple $\Pi = \langle V, I, O, \gamma \rangle$ where

- V is a finite set of **propositional state variables**,
- I is an interpretation of V called the **initial state**,
- O is a finite set of **operators** over V , and
- γ is a formula over V called the **goal**.

Running Example: Planning Task

Example

From the previous chapter, we see that the running example can be represented by the task $\Pi = \langle V, I, O, \gamma \rangle$ with

- $V = \{i, w, t_1, t_2\}$
- $I = \{i \mapsto \mathbf{F}, w \mapsto \mathbf{T}, t_1 \mapsto \mathbf{F}, t_2 \mapsto \mathbf{F}\}$
- $O = \{m_1, m_2, l_1, l_2, u\}$ where
 - $m_1 = \langle \top, ((t_1 \triangleright \neg t_1) \wedge (\neg t_1 \triangleright t_1)), 5 \rangle$
 - $m_2 = \langle \top, ((t_2 \triangleright \neg t_2) \wedge (\neg t_2 \triangleright t_2)), 5 \rangle$
 - $l_1 = \langle \neg i \wedge (w \leftrightarrow t_1), (i \wedge w), 1 \rangle$
 - $l_2 = \langle \neg i \wedge (w \leftrightarrow t_2), (i \wedge \neg w), 1 \rangle$
 - $u = \langle i, \neg i \wedge (w \triangleright ((t_1 \triangleright w) \wedge (\neg t_1 \triangleright \neg w)))$
 $\quad \quad \quad \wedge (\neg w \triangleright ((t_2 \triangleright w) \wedge (\neg t_2 \triangleright \neg w))), 1 \rangle$
- $\gamma = \neg i \wedge \neg w$

Mapping Planning Tasks to Transition Systems

Definition (Transition System Induced by a Planning Task)

The planning task $\Pi = \langle V, I, O, \gamma \rangle$ **induces** the transition system $\mathcal{T}(\Pi) = \langle S, L, c, T, s_0, S_\star \rangle$, where

- S is the set of all states over V ,
- L is the set of operators O ,
- $c(o) = \text{cost}(o)$ for all operators $o \in O$,
- $T = \{ \langle s, o, s' \rangle \mid s \in S, o \text{ applicable in } s, s' = s[o] \}$,
- $s_0 = I$, and
- $S_\star = \{ s \in S \mid s \models \gamma \}$.

Planning Tasks: Terminology

- Terminology for transitions systems is also applied to the planning tasks Π that induce them.
- For example, when we speak of the **states of Π** , we mean the states of $\mathcal{T}(\Pi)$.
- A sequence of operators that forms a solution of $\mathcal{T}(\Pi)$ is called a **plan** of Π .

Satisficing and Optimal Planning

By **planning**, we mean the following two algorithmic problems:

Definition (Satisficing Planning)

Given: a planning task Π

Output: a plan for Π , or **unsolvable** if no plan for Π exists

Definition (Optimal Planning)

Given: a planning task Π

Output: a plan for Π with minimal cost among all plans for Π ,
or **unsolvable** if no plan for Π exists

Summary

Summary

- **Planning tasks** compactly represent transition systems and are suitable as inputs for planning algorithms.
- A planning task consists of a set of **state variables** and an **initial state**, **operators** and **goal** over these state variables.
- We gave **formal definitions** for these concepts.
- In **satisficing planning**, we must find a solution for a planning task (or show that no solution exists).
- In **optimal planning**, we must additionally guarantee that generated solutions are of minimal cost.